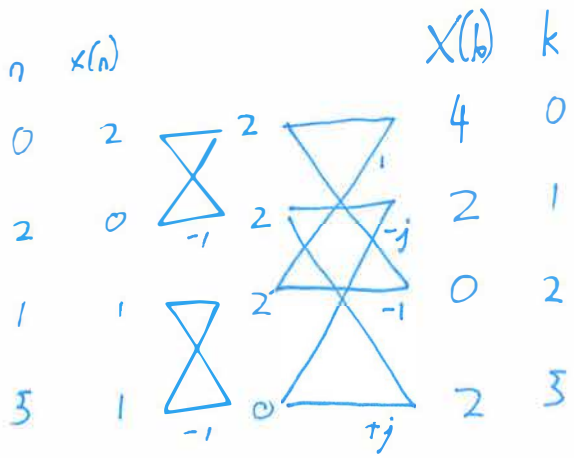
Name Answers

EE-3221 – Dr. Durant – Quiz 8
Winter 2020-'21, Week 10

This is an ***open***-book quiz. Open notes. You may use a calculator.

1. (6 points) Use the FFT algorithm from class (radix-2 decimation-in-time FFT) to calculate the 4-point DFT of $x(n) = 1 + \cos\left(\frac{\pi}{2}n\right)$ = [2 1 0 1]



$X = [4 \quad 2 \quad 0 \quad 2]$

Observations:
- Sum of cosines/even signals → Real DFT
- DC value of 1 → $1 \cdot N = 4 = X_0$ in DFT
- The cosine is at the $k=1$ frequency. $\Omega = \frac{2\pi}{N} \cdot k = \frac{2\pi}{4} \cdot 1 = \frac{\pi}{2}$

  Cosine amplitude=1 ∴ $1 \cdot N = 4$ in DFT, split between $k=1$ & $k=-1 \equiv 3$

  ↑ DFT is periodic due to time-domain sampling

(turn over for 2nd problem)

2. (4 points) x(n) has length 1536 ($2^{11} - 2^9$) and h(n) has length 512 (2^9). By the width property, the convolution has length 2047 (2^{11}-1). Calculating the convolution directly would require 1536 × 512 = 786,432 multiplies.
   a. List the 4 steps to calculate the convolution result using radix-2 (thus N = 2^k) FFTs and IFFTs (inverse FFTs). Hint: Convolution property of DTFT: $Y(e^{j\Omega}) = H(e^{j\Omega})X(e^{j\Omega})$
   b. Calculate the number of (complex) multiplies needed by each step. Recall that a radix-2 FFT has N inputs and k layers, therefore it requires kN multiplies. You do not need to treat real multiplies differently or account for trivial multiplies (by 1, j, etc.).
   c. Add up the multiplies needed and calculate how much the required computing power due to multiplies has been reduced by using the FFT method.

(a) Need DFT length ≥ 2047 to avoid temporal aliasing (wrap-around).
Use $N = 2048 = 2^{11}$

① Take 2048-pt FFT of x after 0-padding (512 0s)
② " " " " " h " " (1536 0s)
③ Multiply the 2 DFTs
④ Inverse DFT of result

(b) ① $k \cdot N = 11 \cdot 2048 = 22,528$
② " $= 22,528$
③ $N = 2,048$
④ $k \cdot N = 22,528$

(c) $\mathcal{E} = 69,632$

$1 - \frac{69,632}{786,432} = 91.15\%$ savings

Note: Beyond efficiencies mentioned in 2b, an FFT that knows its input is 0-padded can eliminate many more multiplies. So, we have a lower bound on the savings.